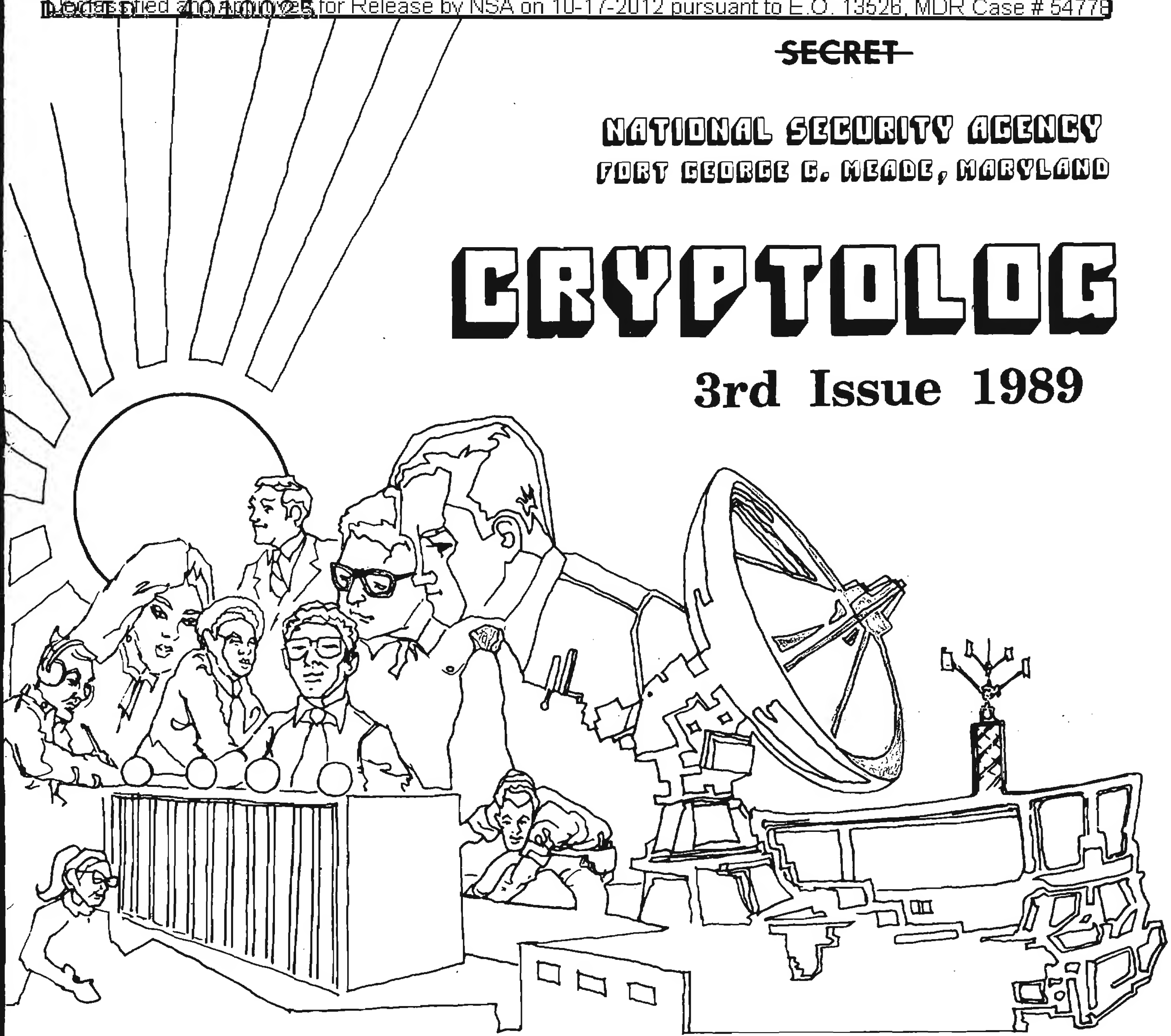
NATIONAL SECURITY AGENCY
FORT GEORGE G. MEADE, MARYLAND

CRYPTOLOG

3rd Issue 1989



50 YEARS WITH UNCLE SAM 1
ERRATUM 6
MAPS AS VITAL RECORDS . . . . [redacted] . . 7
DINOSAURS ARE ALIVE AND WELL. . . . [redacted] . . 9
PORTABILITY PROBLEMS IN C PROGRAMMING . . . [redacted] .12
SPECIAL INTEREST GROUP ON FUSION. 17
COGITATIONS OF A CONTUMACIOUS CABALIST. . . [redacted] .18
LETTERS 19
ANSWERS TO TA EXERCISE. 20
HARDWARE REVIEW: THE NeXT COMPUTER [redacted] .21
BULLETIN BOARD. 28
REVIEW: STRUCTURE AND INTERPRETATION OF COMPUTER PROGRAMS . . . [redacted] .29
CLA ESSAY CONTEST 32
TA EXERCISE . . . . [redacted] .33

P.L. 86-36

DOCID: 4010025

P.L. 86-36

CRYPTOLOG

Published by P1, Techniques and Standards

VOL. XVI, No. 3. . . . . .3rd Issue 1989

To submit articles or letters by mail, send to:
Editor, CRYPTOLOG, P1, NORTH 2N018

If you used a word processor, please include the mag card, floppy or diskette along with your hard copy, with a notation as to what equipment, operating system, and software you used.

via PLATFORM mail, send to:
cryptlg@bar1c05
(bar-one-c-zero-five)
(note: no 'o')

via ALLIANCE, send to:
PLBROWN [note: all caps]
attn: CRYPTOLOG

Always include your full name, organization, and secure phone; also building and room numbers.

For Change of Address
mail name and old and new organizations to:
Editor, CRYPTOLOG, P1, NORTH 2N018
Please do not phone.

ABOUT DISTRIBUTION

We had been submitting address changes to Distribution at the eleventh hour, just before CRYPTOLOG went to press, in an effort to keep up with the last-minute moves of subscribers. But ironically, it turns out that the Press greatly improved its turn-around time, outpacing that of Distribution, so the changes were entered well *after* the eleventh hour And that's why we've been having problems with distribution, except for times when CRYPTOLOG was held up at the Press.

So now we have to go the way of commercial magazines ("allow six to eight weeks for change of address . . .") and close the distribution list well before we're ready to print.

While we're on the subject of distribution, let us direct your attention to a paragraph in very small print that appears elsewhere on this page. It reads,

Contents of CRYPTOLOG may not be reproduced or disseminated outside the National Security Agency without the permission of the Publisher. Inquiries regarding reproduction and dissemination should be directed to the Editor.

This means that

- copies of CRYPTOLOG may not be forwarded to subscribers in the field.

- field stations may not pass copies of CRYPTOLOG to their subordinate field entities.

For further information about the distribution of CRYPTOLOG to the field, please write or call the Editor.

[illegible]

~~FOR OFFICIAL USE ONLY~~

50 YEARS WITH UNCLE SAM

~~(FOUO)~~ *The popular perception of a federal employee with 50 years of service is that of a timid, colorless person shuffling papers in an obscure corner of an obsolete government bureau. The protagonist of this article is no such thing. He is a visionary, NSA's own Jules Verne and Arthur C. Clarke, whose imagination was lauded by NSA's first Deputy Director. He is a man of integrity, an activist who speaks out when moved to do so. He is a doer who actually gets out in the field to implement his theories. He is a world-class scientist, unknown to PBS audiences only because his endeavors, and his association with them, are classified. Among his many honors is his election to the prestigious Cosmos Club for original research and publications on the ionosphere.*

(U) During an exceptional career, Nate, through his energy, enthusiasm, and dedication to solving the real problems, has inspired others and broken new ground. He has won the gratitude, affection, and respect of his colleagues here and abroad. He is the epitome of a visionary, scientist, and explorer.

(U) This tribute was composed by his friends and admirers, and submitted with their very great affection and respect.

(U) When young N. C. (Nate) Gerson returned from summer training with the Massachusetts National Guard in 1934, he found a telegram from the Government Printing Office in Washington, offering a position at $90 a month. The country was in the grips of the Great Depression and jobs were unavailable, so his mother was convinced that President Roosevelt personally made the selection to help the family. Thus started a federal career that stretched over half a century.

(U) In 1938 he transferred to the US Weather Bureau and subsequently was assigned to San Juan, Puerto Rico. There he worked all night (2200-0600) and attended the university as a full time day student (0800-1600). After three years with four hours sleep a day he graduated magna cum laude with a BS in physics. Then he began what was to be a series of "firsts." He completed the first thorough evaluations of pseudo-adiabatic equations. In 1943 in Puerto Rico he made the first upper air wind determinations using wartime radar and resonant dipoles suspended below radiosonde balloons. This technique was implemented for routine observations of upper air winds. He also designed all the thermodynamic charts needed to evaluate upper air soundings, many of which are still in use today by the Weather Bureau and by DoD. This work was not without peril, as Nate reports:

While working at Arecibo, the open cable car taking me from the carriage house, (500 ft above the antenna) to the ground struck the feed structure. The car "caught" and became horizontal as it was haltingly dragged over the structure. The car almost dislodged itself from its support cable; had it done so it would have plunged to the ground. I hugged the only vertical (now horizontal) strut on the flimsy cable car for dear life. After scraping clear, the car swung wildly while its support cable oscillated vertically 15-20 feet — all about 500 feet in the air. I was glad to reach the ground. A $10 override switch in the circuitry would have prevented a mid-air collision course between the cable car and the feed structure. I expressed my peevishness to the management.

(U) In 1945 Nate met a beautiful creature in the WAVES. She married him before realizing that his annual salary was only $1620. The Navy, during their honeymoon, assigned her to Barracks D with 3999 other women. After their return to Washington, Nate would visit her in the evening until at curfew an enormous WAVE wearing an SP armband would tap him on the shoulder and say, "Time to go home sonny."

(U) One day in 1946 an old professor sauntered into the office and told Nate's boss that he wanted Nate to work for the Army Air Corps. Listening with apprehension Nate learned for the first time that his boss appreciated him, but the professor won, and Nate was shifted to the US Army Air Corps' Watson Laboratories at Fort Monmouth, NJ. He left Washington by train on a Monday morning, telling his wife he would return by Friday. Thursday morning he was given 120-day travel orders, later extended another 120 days. He crisscrossed the arctic from Hudson's Bay to Alaska and visited (under primitive conditions) more of the Canadian north than most Canadians ever will see.

(U) At the Watson Labs Nate assisted in evaluations of 180 kHz experimental Loran navigational systems in southwestern Canada, later implemented as LF LORAN. He was responsible for its calibration and acceptable operation at all field sites in Canada and Alaska, from 45° N to beyond the Arctic Circle. Its reliability led to its installation globally as an electronic navigational system for ships and aircraft. And another first: Nate completed the first analysis of natural LF noise intensities in the subarctic.

(U) While doing all of this, Nate managed to obtain an MS in physics from New York University, and has worked in that field ever since, first in thermodynamics then in electrodynamics.

(U) In 1948 the Army Air Corps split from the Army and became an independent Service, the Air Force. Nate was asked to organize a research laboratory devoted to radio wave propagation. Feeling that knowledge of the medium (the ionosphere and atmosphere) was essential, he established the Ionospheric Physics Laboratory, still in existence, and became its Chief and one of the five founders of the Air Force Cambridge Research Center near Boston. Characteristically, he took charge at once, and reoriented the USAF radio propagation research from statistical studies of intensity variations on point-to-point circuits to an understanding of the physics and dynamics of the ionosphere. Among his many initiatives during this period was the implementation of a follow-the-sun (as the earth rotated) airborne observatory to obtain ionospheric soundings over the North Pole; it then became possible to examine arctic ionospheric drifts and to obtain midlatitude airglow spectra.

(U) Early on, Nate revealed his mettle as innovator and goad, backed by that single-minded purposefulness that is an elemental factor of success. Between 1948 and 1956 he made pioneering studies of auroral, ionospheric, and geomagnetic physics and prodded the U.S. and Canada into becoming world leaders in this field. He won over initial opposition, and instrumented an airborne laboratory for ionospheric and spectrographic measurements, a concept later copied by the Soviets. The plane he sent to the arctic developed mechanical problems, drastically reducing the flying hours over the North Pole, which led him to propose and justify the US station at the South Pole. (In sun-earth geometry the poles provide unique observational points.)

~~(FOUO)~~ In 1956 he was invited to present a lecture on ionospheric winds at Arlington Hall Station to a group unknown to him — the newly formed National Security Agency. Before the meeting ended he was asked to consider a consultantship. Shortly afterwards the US National Committee for the International Geophysical Year suggested that he join Deep Freeze II, the US Antarctic Expedition, as Admiral Byrd's Chief Scientist. However, by that time he had become hooked on the challenges facing NSA, and he became a full-time employee in the old REMP—Research, Engineering, Math and Physics.

~~(FOUO)~~ Nate's insight into NSA's problems has been extraordinary. This probably stems from his stint with the Air Force, where he looked at propagation from the point of view of communications; at NSA ,the viewpoint is that of intercepting. Now one of his primary concerns is improving the front end of the SIGINT system — intercept.

~~(C)~~ And as some of you know all too well, Nate speaks his mind, even when his comments might be unwelcome. For example, he discounted rosy estimates [redacted] Unfortunately he was proven right —the physics just did not permit it.

~~(S-CCO)~~ The range of his activities at NSA is astounding. Among his many undertakings, he examined High Frequency Direction Finding accuracies and system improvements, intercept from HF to SHF, general search, and electromagnetic wave propagation in general. [redacted]

Nate relates his adventures ...

The Roof and Floor of the Earth

In Antarctica I slipped on an icy hillside and slid 3000 feet free-fall to the bottom. On another occasion my Weasel stalled on a very steep icy path. I parked and carried the 750-pound dummy load cross country uphill 200 yards through knee-deep ice-encrusted snow. The flight "to the ice" from New Zealand was unusual; a forward hatch unexpectedly opened and discharged a life raft which automatically inflated and slammed against the tail section. The plane was thrown out of control until the life raft finally blew off. Unfastened passengers were thrown about like corks. I had been sleeping on the cargo and could not reach a bucket seat until grabbed by friendly hands. The pilot recovered less than 1000 feet above the sea. We were three hours from McMurdo Sound.

In the Arctic one December morning when I left the mess hall after breakfast for work, I was blown down onto the snow four times during the 3/4 mile walk to the Operations Building. Time for the "walk" was 45 minutes. During two summers (outdoor temperature +52F) the work was so arduous that perspiration soaked through all my clothes to the outside of my parka. On several trips I lost 10 pounds during the first week. While I was working at isolated outside facilities one summer, grazing caribou came within 15 feet. On one occasion, working alone far from the station, I came across fresh wolf tracks in the soft tundra.

[redacted]

EO 1.4.(c)
P.L. 86-36

EO 1.4.(c)
P.L. 86-36

EO 1.4.(d)
P.L. 86-36

~~SECRET~~

[redacted]

Two of the many congratulatory messages sent to Nate

~~SECRET~~

~~HANDLE VIA COMINT CHANNELS ONLY~~

DOCID: 4010025

EO 1.4.(d)
P.L. 86-36

[redacted]

[redacted]

~~(C)~~ At NSA Nate has bounced between DDR and DDO, alternating between theoretical science and practical applications, as he prefers to do. He has regularly visited field sites to gain first hand understanding of the operational system as it is, not as it is surmised to be. He spent over four years beyond the polar circles, and is the only Agency employee sent to both the Antarctic and the Arctic.

(U) Like the professor he might have been, he is adamant that every project produce a positive or negative final report in good idiomatic English to guide future workers, "otherwise you flunk the course." Moreover, the reports must be short, and to the point. Some unhappy souls discovered, too late, that Nate has a short fuse for six-pound reports containing only one sentence of value.

~~(FOUO)~~ High on his wish list for NSA is that quality outweigh quantity; a close second is that managers be more aggressive in sifting wheat from chaff. He urges Engineering to be somewhat more responsive to Operations, but he would like to see Research given higher internal status (and set higher goals). At the same time, he insists that proposals be more critically evaluated before adoption. As for people, the single most valuable attribute anyone can have is integrity.

(U) Still a starry-eyed idealist, he believes (even as he scolds it) that R&E is comparable to the Lake Woebegone Community: all the women are beautiful, all the men dedicated, and all the Elders wise.

P.L. 86-36

ERRATUM

In 2nd Issue 1989, "Compusec Policy for the Intelligence Community," by [redacted] *two captions were inadvertently transposed. Please replace the published diagram with the one shown below. CRYPTOLOG regrets the error.*

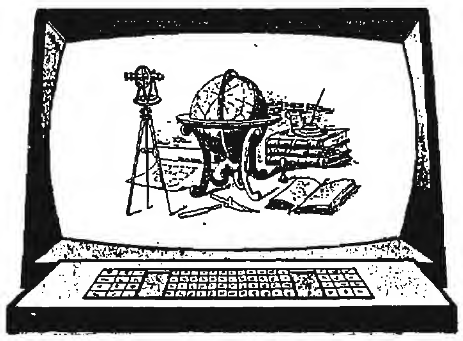
MODES OF OPERATION

USER ATTRIBUTES	Dedicated	System-high	Compartmented	Multilevel
Need-to-know	All	Some	Some	Some
Access	All	All	Some	Some
Clearance	All	All	All	Some

unclassified

DOCID: 4010025

MAPS AS VITAL RECORDS



[redacted] *T541*

P.L. 86-36

(U) Two aspects of the emergency preparedness programs were prime factors in the revitalization of the Vital Records program. These were (a) having records available which will ensure the effective conduct of essential United States Government activities, and (b) having sufficient means, people, equipment and physical plant to manipulate these records as needed.

~~(FOUO)~~ The revitalization began in October 1985 with a memo from the then NSA Records Officer, [redacted] reminding all Agency elements that the Director had tasked the Agency with designing and building "a framework for a survivable SIGINT system, under all conditions, including general war." Through seminars and personal contacts, the Senior Key Component Records Officers were apprised of the T54 plans. With the assignment of a vital records controller to T5411, planning began in earnest.

(U) The element records officers analyzed their element's operational needs and identified the records that they considered vital to their survivability and reconstruction plans. Among them were the map requirements. These were submitted to the controller.

(U) After analyzing the requirements for maps and geographic aids submitted by the various records officers, the controller noted that several components had a requirement for the same materials. And just at that time came an evaluation of the Agency's performance during the Joint Chiefs of Staff exercise PRESENT ARMS, which concluded that maps and geographic aids would have greatly assisted the NSA participants in performing their tasks. The Controller decided to develop a single file to make all map and related information available to all possible users. Thus was born the idea of the Common User Module MAPNEED (CUMMN).

(U) Several things had to be done before the maps and geographic aids could be indexed into a retrievable data file, then shipped to and stored at a Vital Records Depository. First was to develop a system of accounting for each item placed into the data file, as well as a manual retrieval system. Then all maps had to be folded to fit the legal size folders used for shipment and storage. The folders were marked with identifying information as depicted in Figure 1, then placed in boxes marked CUMMN ##.:

Chart	Country (ies)
Sheet #	Placename
Sheet #	Placename

Figure 1

~~(FOUO)~~ The controller and the T54 Project Officer for Archival Processing developed a file containing a collection of maps and geographic aids that were identified as vital to the Agency teams engaged in reconstruction, emergency relocation, or critical post-disaster operations. This was provided to the element records officers.

~~(FOUO)~~ The file took form as maps and geographic aids, procured through the T54 Map and Geographic Aids Library, began to arrive. It includes a machine database called MAPNEED with a backup file called NEWMAP. Both the original database and the backup file use the dBASE III PLUS system and show identifying information about the maps and geographic aids contained in the file, as well as other identifying data. As items that are out-of-stock or out-of-print are procured, the file will increase in size. 5,044 items are currently indexed into the data file.

(U) The procedure is virtually self-evident:

- Enter the data into each field and press (CR).

~~FOR OFFICIAL USE ONLY~~

```
D B A S E   I I I   P L U S
   S T A R T U P   M E N U

1. Use NARADEX Programs
2. Use ANSUM Programs
3. Use ACCONLOG Programs
4. Use Vital Records Programs

0. EXIT
```

Step 1. Select 4.

```
V I T A L   R E C O R D S

1.  YAKIMA Records
2.  MAPNEED Records

0.  EXIT
```

Step 2. Select 2.

```
M A P N E E D

1. ADD INFORMATION
2. CHANGE INFORMATION
3. REMOVE INFORMATION
4. ADD COUNTRY INFORMATION
5. PINT OUT FILE

0. EXIT
```

Step 3. Select desired action.

~~FOUO)~~

Figure 2. Accessing the data base

▸ When all data has been entered, press (PgDn) to move to the next record.

▸ Press (Ctrl/End) to save all the new information that has been entered

▸ When the computer screen indicates 0 (select 0) press (CR) until a dot prompt appears.

▸ At the dot prompt, type <`USE MAPNEED INDEX MAPCHART`> and press (CR),

▸ Type <`REINDEX`> and press (CR). This command reindexes the file and place the new record containing the new information in its proper place.

Field		Type	Length
1	REC_NO	C	6
2	CHART	C	50
3	PROJECTION	C	20
4	SCALE	C	18
5	LOC_INFO	C	80
6	BOX_NR	C	7
7	ORG	C	20
8	COMMENTS	C	50

Explanatory Notes

1. REC_NO indicates the record number.
2. CHART is the short title of the specific chart otherwise the long title.
3. PROJECTION indicates the type of projection requested by the user if available.
4. SCALE is the ratio of units used on the map or chart, i.e., 1:50,000.
5. LOC_INFO is the country or geographic area covered by the map or chart, or in the case of UTM charts, the long title.
6. BOX_NR refers to storage
7. ORG is the organizational designator(s) of the requesting NSA organization. Multiple designators are separated by backslash (\).
8. COMMENTS additional descriptive data

(U)

Fig. 3. Data Structure of MAPNEED and NEWMAP

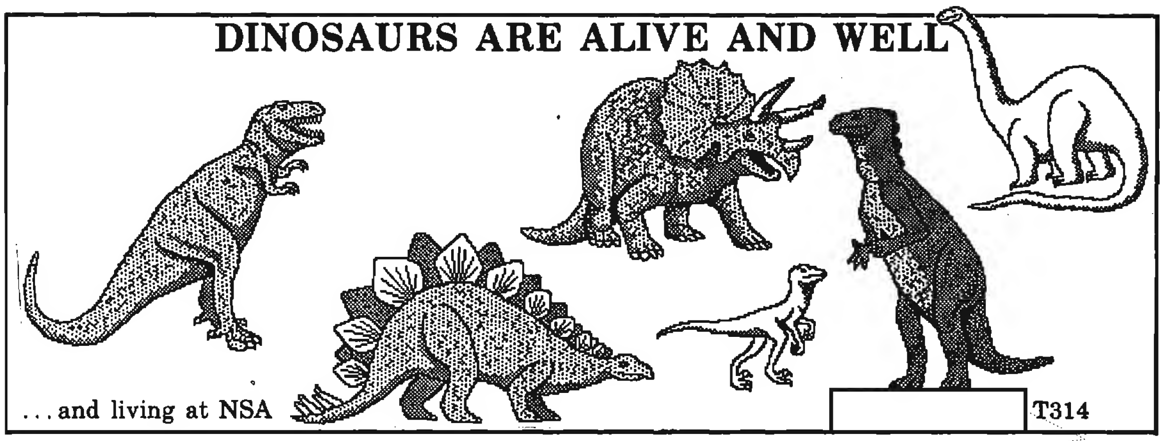
(U) All the routines for modifying and accessioning data are just as easy to do. It is possible to retrieve information on any the fields shown and to print it out. There are procedures for selecting all charts required by a specific organization or user, or by a combination of them. It is even possible to make a "gang" selection to include all organizations subordinate to the one cited. ▫

CRYPTOLOG

welcomes your letters and comments

~~FOR OFFICIAL USE ONLY~~

DINOSAURS ARE ALIVE AND WELL



...and living at NSA [redacted] T314

P.L. 86-36

The article "Software Acquisition: A Reform in Need of Reform" by [redacted] 2nd Issue of 1988, prompted me to propose a radically different view from the one proposed in that paper. I contend that software acquisition is not so much in need of reform as are the philosophies and practices of those who manage the software acquisition or development. Seat-of-the-pants software acquisition and development didn't die with the advent of formalized procedures. It's thriving at NSA, kept alive by archaic software management practices and philosophies that have failed to adapt to the new generation of concepts in the software world.

IN-HOUSE SOFTWARE DEVELOPMENT

Poor management is the main reason for NSA's software development problems. This includes inept Software Development Managers who fail to plan and organize the development effort, the laissez-faire style of upper management that fails to enforce commitments and has a "hear no evil, see no evil" attitude, and apathetic external organizations with no active interest in software development.

THE CUSTOMER

The road to failure doesn't begin with the development organization or the program management office but with the customers' lack of committed involvement and poorly defined and expressed needs. Only when the system is deployed do they take an active interest in the product. The pervasive attitude seems to be "Here, Research and Development, perform a miracle with this sparse information and we'll talk to you when it's done or when we need a change in the miracle."

While overstated, this situation is real. At this juncture, the customer is the only one who knows specifically what it needs and what information is necessary to meet those needs. Operational organizations should place greater emphasis on conveying those needs as clearly defined objectives and requirements. Often requirements are so generic that what takes place is "interpretation by misperception," (i.e. a mismatch between what the developer understands and what the customer meant). Continuous customer involvement during the entire development process is vital to clear, concise requirements. The technical exchange and direction the customer could provide would help immensely in guiding an errant project.

UPPER MANAGEMENT

The most frustrating obstacle to overcome is upper management's non-receptiveness to issues concerning suspect software development practices. "You can knock forever on a deaf man's door," says Zorba the Greek. The message they're sending is interpreted as non-support and indicates a lack of trust in their subordinates. This action is hypocritical for an organization espousing the benefits of NSA manuals on software development. One wonders from what perspective upper management is viewing the situation.

One could postulate that the motives are partially financial. The measure of an organization's worth has become the size of its budget, instead of its efficiency and effectiveness. And since the organization is budget-based, the normal business definitions for "performance" and "results" have changed. "Performance" becomes the ability to maintain or increase the organizational budget, while "results" is the acquisition of a larger budget (Drucker). By accepting troubled systems, the worse off the better, management acquires justification for their budget. Unfortunately, it's justification by non-performance.

Another perspective, using 80's terminology, involves male bonding and the "good ol' boy" upper management fraternity where shunning inter-organizational conflict is all too prevalent. Conflict is inevitable and, in my opinion, necessary in our dynamic working environment. Software maintenance organizations have been compromising their ability to maintain newly developed software because of upper management's resistance or inability to take stands on software development issues that cut across organizational boundaries. It is amazing that a stance has not been taken in light of some of the recent software disasters.

Upper management difficulties are not solely internal to the organization: problems are also generated by external organizations failing to acknowledge and enforce the responsibilities they approved. Management's concurrence on documents has become more of an exercise in form than in substance. Corrective measures are rarely taken when identified obligations are not fulfilled. There is a definite truism in relating management signatures to the cliche "... not being worth the paper it is written on".

THE DEVELOPMENT ORGANIZATION

The lack of proper management practices is nowhere more apparent than on projects with software development problems. The use of formalized software development techniques isn't the root of the problem since in most instances they are not being used anyway. The problem lies in the inconsistent or nonexistent application of basic management principles: planning, organizing, integrating, and evaluating. Formalized software development techniques are nothing more than an advanced application of these basics.

Planning, as I loosely refer to it, is usually done in one of two ways. The first is planning by mental imagery. The software development or acquisition manager knows what the overall objectives are and begins to visualize and initiate the steps needed to get the ball rolling. He then applies the same technique to a variety of sub-objectives. Eventually, he is overwhelmed trying to keep track mentally of everything going on. So the project that started out with the best of intentions falls into disarray, missing schedules and going over budget, and turns into a QRC effort - not a Quick Reaction Capability, but a Quick Reaction to Chaotic conditions. The two are similar, in that both are undocumented, untested, and meet only the minimum requirements.

Planning by camouflage is used when a boilerplate Software Development Plan is written. The document is written in a half-hearted attempt to satisfy an NSA reg but is not periodically updated. All the right buzz words are there, but unfortunately the document lacks substance. Nevertheless, the document is waved like a banner in proclamation of good software development techniques. Eventually, its fragile nature becomes evident as the complexity of software development increases. Development collapses because of a false sense of security in having a plan, owing to a common but mistaken belief that planning is a one-time event rather than an on-going activity.

Without adequate planning it is unfair to discuss the organization and integration of project resources. Whether the resources are time, budget, personnel, facilities, or managerial style, success is directly related to the effectiveness of planning. On problem projects, the software manager is in a reactive rather initiative mode when it comes to coordinating and utilizing resources. In the past, it was possible to compensate for poor planning by drawing on additional resources, for even if the project came in late and over budget, -- who cared -- the money was there. As we move into a period of greater austerity, this avenue of escape will no longer exist. Poor project management will become increasingly apparent.

More often than not, development organizations become afflicted with software development amnesia, so mistakes made in previous projects recur, usually in the form of schedule slippages and over-expenditures. Critical evaluation of a project isn't something that is done just at project completion; like planning, it is an on-going activity. Evaluation of mistakes provides an opportunity to learn how not to do things as well as how to do them.

SOFTWARE ACQUISITION BY CONTRACTING

Is software acquisition more expensive than in-house software development? Not really. There are hidden costs in in-house software development owing to large adjustments in schedules and our hypocritical attitude toward documentation. When software is developed in-house, schedule slippages lock development personnel in place, adding to the initial cost and delaying the opportunity to begin new projects. Adding to that cost is documentation that is non-existent or poor in quality. In most cases, software documents have to be upgraded to a level of acceptability by the life cycle support organization using their own personnel or contracting the work out.

Acquisition is not a panacea for the Agency's software development problems. I do not advocate contracted efforts and do not believe the end product is any better just because it was developed by a contractor. What I have found is that acquired systems are better documented and experience fewer schedule slippages. The reasons for this are contractual and financial. Software documents are identified as part of the Contract Data Requirements List and the contractor is motivated by profit to avoid excessive schedule slippages. Problems surface when the Agency provides inadequate requirements definition in the purchase description or asks for modifications because of unforeseen environmental changes. Of course the contractor is going to ask for more time and money. These risks should already have been considered when opting to acquire, rather than develop, new software.

RECOMMENDATIONS

▸ Don't blame Agency software development practices and procedures for the project's failures. "It is a poor workman that blames his tools for shoddy work." Instead, tailor them to meet the size and needs of the project. In this case, familiarity breeds proficiency.

▸ Try to obtain the clearest possible definition of what the customers really need. The customers should be as involved in the project as is the developer. If they aren't, they have only themselves to blame for the type of system delivered.

▸ Become receptive to bad news during software development. Not only must you listen to what is being said, ou must act. A Janus-faced stance on enforcement of Agency standard software development practices promotes organizational inefficiency and ineffectiveness.

▸ Accept and use the Agency's software development methodology. Credibility to deliver acceptable software is being undermined by resistance to using new techniques. The track record for previously developed software indicates changes are needed. There are reasons for the existence of the NSA regs.

▸ Provide more opportunities to software support organizations to develop software. Presently, by charter, most of the Agency's new development is being done by research and development organizations. Expanding development opportunities will give the customers a choice of developers and make the process. Quality then becomes the prime justification for increasing organizational budgets. The customers usually keep selecting a developer whose projects not only perform, but are timely and cost-effective. The Agency should no longer keep all of its development eggs in one basket.

CONCLUSION

My opinions are based on experiences and observations in software support. My concern is not for present but for future developments. In most cases, it is too late for constructive suggestions to have an impact on current projects. Even the simple identification of problems is often seen by the developer as meddlesome intrusions. Changes need to be made in the way the Agency handles future software development. The Agency is too dynamic and its software personnel too professional to be sabotaged by archaic software development philosophies and practices. If the Agency fails to adapt to the new software environment, it runs the risk of meeting the same fate as the dinosaurs. Those large entities were overcome by their failure to adapt. Unfortunately, too many Agency dinosaurs are still developing software.

ACKNOWEDGEMENTS

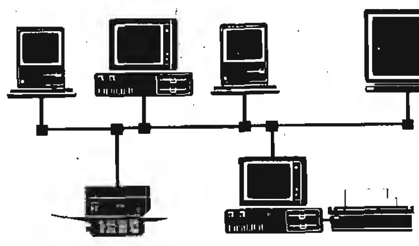
My thanks to [redacted] T31 Technical Writer, for her constructive comments.

P.L. 86-36

~~FOR OFFICIAL USE ONLY~~

Portability Problems in C Programming

. . . and how to avoid them



[redacted]

P.L. 86-36

Many portability problems are hidden from the C programmer because C is a high-level language. Some of the flexibility which allows C to be easily implemented on so many machines also allows for variations in a number of well-defined ways. A programmer must be aware of these variations in order to be able to write portable code. The most important variations are described below.

WORD SIZE

The common word sizes today are 16 and 32 bits, with some 8-bit words on aging micro computers and 64-bit words on some super computers. In C/UNIX the common word size porting problem comes from using the variable type "int". An int is defined as the storage class that is most efficient on the given machine. On a 16-bit machine an int is 16 bits; on a 32-bit machine it is 32 bits.

Problems arise when you try to go from larger to smaller word sizes and find that an int is no longer large enough to hold your information. Problems also arise when you try to write data in the form of an int to a disk file. On a 16-bit machine it will take 2 bytes of disk space; on a 32-bit machine it will take 4 bytes. If a program believes it knows how many bytes of data are on the disk it may be surprised when compiled on a different machine.

For portability one should use the variable type "char" for 8-bit data, "short" for 16-bit data, and "long" for data which is 32 bits or more in size. Remember too that an unsigned quantity will allow a magnitude twice as large as the original.

Some examples of different sizes :

```
short=2, int=2, long=4     :  PDP-11, AT (XENIX)

short=2, int=4, long=4      :  386(SCO-XENIX), SUN-3-4, ASH(3b15/20), CONVEX

short=3/4, int=8, long=8   :  CRAY
```

Further, programs should use the "sizeof" compiler construct if they need to reference the size of an "int" variable. Unfortunately there is a lot of code that doesn't use these more portable constructs.

BYTE ORDER

Different machines have different ways of storing 16- and 32-bit quantities. Let's first consider the ways a 16-bit value might be stored :

```
short variable = 0x0102;    /* 0x is notation for hex */
```

this may be stored internally as

```
01 02            /* each 2 digit number is */
   or            /* is a byte              */
02 01
```

Now consider the ways in which a 32-bit value might be stored :

```
long variable = 0x01020304;   /* 0x is notation for hex */
```

this may be stored internally as

```
01 02 03 04         /* each 2 digit number is */
    or              /* is a byte              */
04 03 02 01
    or
02 01 04 03.
```

There are two kinds of portability issues that arise from this difference.

The first problem comes from using more than one kind of storage class to access this information. For instance, suppose you allow a short (16 bits) to represent a character on a display screen. In the short the Most Significant Byte is the actual character displayed and the Least Significant Byte is its attributes (foreground color, background color, blink, underline, etc.). When you initially put a character in this short you will store both the character and the attribute. Now suppose that you wish to write over the character but not affect the attribute. Most programs will let a character pointer point to this short and then change the first byte. This is not portable since it only works on machines with the first byte ordering scheme. If the second scheme is used the MSB is actually the second byte. This problem is even worse when dealing with 32-bit quantities since there are more possible outcomes.

The second kind of portability issue deals with writing this data to a disk file. A file that is created by writing shorts or longs to the disk will not have the same meaning if it is transferred to a different machine with a different byte order. This gives rise to interoperability problems between machines and the need to translate structured data files when transferring from machine to machine.

Some examples of variants

```
with 8 bytes in order being = 0102030405060708
2-byte short=0201    4-byte long=02010403    : PDP-11
2-byte short=0201    4-byte long=04030201    : AT(XENIX)   386(SCO XENIX)
2-byte short=0102    4-byte long=01020304    : SUN-3-4   ASH(3b15/20) CONVEX
3-byte short=060708  8-byte long=0102..0708  : CRAY
```

BYTE ALIGNMENT, STRUCTURE SIZE

Some machines allow a short (16 bits) to be aligned in memory on any byte boundary, others allow a short to be aligned only on even byte boundaries. Further some machines allow a long (32 bits) to be aligned on any byte boundary, others allow a long to be aligned only on even byte boundaries, while still others allow a long to be aligned only on 4 byte boundaries.

In UNIX you can group related variables into a structure. A UNIX structure has as its elements each of the related variables. For example:

```
struct example
{
     char  first_char;      /*  8 bits */
     short first_short;     /* 16 bits */
     long  first_long;      /* 32 bits */
};
```

This defines a structure called "example" that has 3 elements "first_char", "first_short", and "first_long". On a machine that allows alignment on any byte boundary the elements in the structure will have the following data storage:

```
struct example
{
   char  first_char;  /* byte 1                                  */
   short first_short; /* byte 2 and byte 3                       */
   long  first_long;  /* byte 4, byte 5, byte 6, and byte 7 */
};
```

And its size will be 7 bytes. On a machine that allows alignment for shorts and longs on even byte boundaries only the following storage will be in effect:

```
struct example
{
   char  first_char;   /* byte 1                                  */
                       /* byte 2 is unused                        */
   short first_short;  /* byte 3 and byte 4                       */
   long  first_long;   /* byte 5, byte 6, byte 7, and byte 8 */
};
```

And its size will be 8 bytes. On a machine that allows alignment only on 4-byte boundaries the following storage will be in effect:

```
struct example
{
    char  first_char;     /* bytes 1                              */
                          /* bytes 2, 3, and 4 are unused         */
    short first_short;    /* byte 5 and byte 6                    */
                          /* bytes 7 and 8 are unused             */
    long  first_long;     /* bytes 9, 10, 11 and 12               */
};
```

And its size will be 12 bytes.

The problem with this comes when a program makes an assumption about the size of the structure. There is a construct "sizeof()" that will return the size of a structure but not all programs use it. Even if they do you can still get into buffer size problems if a structure gets copied into a buffer and the buffer is large enough to hold the structure on some machines but not on others.

Structure size and alignment also lead to the same problems with access by different storage class mechanisms (ie. character pointers) and representation on disk that the byte order differences lead to.

We looked at some examples of the size of a structure containing a character plus an integer of different types :

```
sizeof( struct { char c; short si; } )
```

~~FOR OFFICIAL USE ONLY~~

```
        sizeof( struct { char c; long  li; } )


sizeof(char+short)=4  sizeof(char+long)=6  : PDP-11  AT(XENIX) SUN-3
sizeof(char+short)=4  sizeof(char+long)=8  : 386  ASH  CONVEX  SUN-4
sizeof(char+short)=16 sizeof(char+long)=16 : CRAY
```

SIGN EXTENSION

In UNIX values can be signed or unsigned. If a value is signed its Most Significant Bit is used as a sign bit. For example an unsigned character can range in values from 0 to 255, while a signed character can range in values from -128 to +127. The same also applies to short (16-bit) and long (32-bit) values.

Some machines will extend the sign bit, propagating it to the left, when a smaller storage class variable is assigned to a larger one (ex. long = short), and others will not. In UNIX this happens more often than you might think. Every time a subroutine is called its calling arguments are pushed on the stack as integers (whose size varies according to the word size of the machine).

The problem arises when comparisons are made between values. If I have a short (16-bits) with a hex 0085 stored in it and a character (8 bits) with a hex 85 stored in it and compare them on a machine that does not sign extend they will be equal. However on a machine that does sign extend they will not be equal because the 85 stored in the character will be converted to a hex FF85 for the comparison. To get around this problem some programs will logically "and" the character value with hex FF. Unfortunately this works on some machines but not on others (the SUN 3 will sign extend the result of the logical and the PDP 11 won't).

The only real way to get around this problem is to use unsigned storage classes when dealing with data that may have the MSB turned on. Most programs have some kind of sign extension non-portability built in to them.

Variations which were found are :

❒ *We looked at whether typecast of small data types to larger types resulted in a sign extend*

```
sign extend all cases : PDP-11  AT(XENIX)  SCO386  SUN-3-4  CONVEX
sign extend short to long but not char to any : ASH(3b15/3b20)  CRAY
```

❒ *We looked at whether an arithmetic right shift sign extends*

```
sign extend short and long       : PDP-11  AT(XENIX)  SCO386  SUN-3-4
sign extend only short, not long : ASH(3b15/3b20)  CRAY
no sign extend on right shift    : CONVEX
```

(no machine sign extends an unsigned quantity)
(PDP doesn't allow for unsigned long, and restricted experiments)
(CRAY converts short->int before shift, so sign extending it first)

FLOATING POINT FORMAT

The format used for internal representation of floating point numbers can vary from machine to machine. This can cause the disk file data transfer problems mentioned above and can also cause loss of precision problems to floating point programs.

CONCLUSION

It is probably impossible to give rules which will guarantee a portable program without restricting the flexibility which a language like C allows you. Some hints which might help are:

(a) prefer to use long/short rather than int, and if the object is not a signed integer quantity the type should probably be unsigned.

(b) always use sizeof() to discover the size of an object

(c) make all your output and input data ASCII readable rather than binary

(d) avoid using multiple representations of the same data, e.g., use of unions and certain means of type-casting.

We note first that some of these hints are trade-offs between portability and efficiency. We also note that the 'lint' program will highlight many potential portability problems. The best way to learn the problems is probably to take some small program you have written and port it to as many machines as you can find. You will avoid many of the pitfalls with your next program.

We note that no two of the machines investigated were identical. The program supplied as an appendix will run differently on each of the eight machines investigated.

APPENDIX

```
char *whatmc()

{
      /*  test one data  */
      union { char clist[8]; short si; long li;} mix;
      /*  test two data  */
      char c=0xff; short si=0xffff; long li=0xffffffff;
      /*  test three data  */
      struct { char c; long li; } combine;

     { int i; for(i=0; i<8; i++) mix.clist[i]='0'+i; }



     /*  Test one, picks out a CRAY and PDP-11  */
     if ((mix.si&0xff)==0x37) return "cray";
     if (mix.li==0x31303332) return "pdp-11";

     /*  Test one, splits the rest into two groups  */
     if (mix.li==0x30313233)
     {

       int i=c; /*  Either SUN or ASH or CONVEX  */
       if (i == 0xff) return "ash";
       if (li>>1 == 0xefffffff) return "convex";
       if (sizeof(combine)==6) return "sun-3";
       return "sun-4";
     }
     else
     {
        /* Either an AT or a 386 machine running SCO XENIX  */ if
        (sizeof(int)==sizeof(short)) return "at";
        return "386";
     }

     /*  Never gets here  */ }

/* end of whatmc()  */
```

COMPUTER & INFORMATION SCIENCES INSTITUTE

Special Interest Group on

DATA FUSION

If you use or write software that uses data from multiple sources to track, identify, or evaluate TARGETS, you are doing DATA FUSION and may benefit from exchanging ideas and software with others.

MEETINGS in EARLY 1990

all held in 9A135, HQS, 0930-1030

Wednesday 24 January

Wednesday 28 February

Friday 23 March

Wednesday 18 April

CLEARANCE LEVEL: Top Secret Codeword
ALL PERSONS WITH GREEN OR GOLD BADGES ARE WELCOME

For more information on the [redacted] Special Interest Group, please call:

[redacted] P13, 963-5868
[redacted] A2, 963-3493
[redacted] A6/T4, 963-4743
[redacted] W46, 963-4702

P.L. 86-36

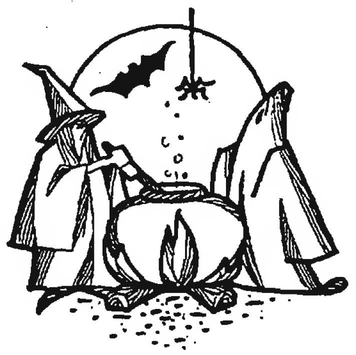
~~FOR OFFICIAL USE ONLY~~

COGITATIONS *of a* CONTUMACIOUS CABALIST

P.L. 86-36

[redacted]

W21



~~(C)~~ Someday it will be recognized as a fundamental truth of DDO that each group deserves the analysts it gets. Conversely, someday it will be admitted that no analyst deserves the group to which he or she is assigned.

~~(C)~~ As a veteran observer of the machinations among DDO organizations, I delight in belonging to a veritable cabal. Throughout the centuries cabals have been seen as covens for witches and warlocks, and as secret organizations from whence no good ever came. That label--CABAL--is not unlike the cloak of mystery which surrounds those who dwell in the heart of W Group.

[redacted]

~~(C)~~ Even to dare pronounce the phrase "W Group" in the same breath as "job" or "position" is not for the faint of heart. For the well-bred sons of DDO (the Groups of A, B, and G) cannot abide the ne'er-do-well son W. In fact several A and G pundits are quite sure that W was not born of the same father at all. They firmly believe that W is an imposter at best, and illegitimate at worst.

~~(C)~~ How can W, the wayward son, compete? To be sure, there is a small linguistic conclave fighting for freedom and the American way as part of W1. But that is a small, elite force. Most of W is not made up of linguists but of engineers, signals analysts, computer scientists and mathematicians (with a few odd intelligence research analysts thrown in to make the concoction odder still).

~~(C)~~ With such an eclectic band of workers, it is no wonder that most people wonder at W group! Try telling the linguist that you study what we affectionately call "green worms." That alone sparks off titters and cat calls. For who hasn't wondered at how the worm gets into the finest tequilla bottles. No sense in telling the linguist it is not that kind of nematode, for as they are wont to say: "worms is worms." No, the plight of the W grouper is worse still because few recognize that ELINT doesn't lie. That is to say, ELINT won't tell you he or she has turned over a new life and resolved to be on the side of truth, justice, and the American way, while still doubling the defense budget. Nay! ELINT just appears out of the sky and bites you when you least expect it. Rather like death, it shows no discrimination.

~~(C)~~ The problem with ELINT is that so few people understand it, which is why we ELINTers are so sorely misunderstood. ELINT is not a class unto itself. It is merely another side of that vixen, SIGINT. Although ELINT is not that scourge known as "voice," it does have melifluous tones and modulations. You might even say it can sing, as it often chirps! It can even illuminate the darkest of hours. As such ELINT is just another chapter in the life of the nefarious vixen, SIGINT. □

P.L. 86-36

P.L. 86-36

Letters

To the Editor:

This is regarding the derogatory and linguistically meaningless comment in [redacted] article on Xhosa (CRYPTOLOG, 3rd Issue 1988) to the effect that Afrikaans is a "bastardized version of 16th century Dutch" (sic):

Afrikaans, as all languages, evolved and developed organically, incorporating numerous lexical and syntactic features of the languages with which it came into contact during its formative period. Afrikaans is, in fact, the only modern European-based language which evolved on non-European soil, adopting many non-Indo-European vocabulary and grammatical features — including those from the Bantu and Khoisan language families, as well as from Portuguese and English. Its beginnings indeed go back to the 16th century, and it was nearly fully developed (with the exception of a standardized orthography) by the early to mid 1700's.

As all languages develop from antecedents, is it accurate to describe English as "bastardized" Danish and Norman French? Italian and "bastardized" Latin? Yiddish as "bastardized" German? Xhosa as a "bastardized" Bantu dialect mixing Hottentot and other non-Bantu elements?

In an article with some linguistic pretensions, I recommend that a modicum of attention be paid to proper linguistic terminology and accuracy, and that the writer's personal prejudices, however keenly felt, be tempered by some knowledge of prevailing linguistic theory in the interest of providing the reader with accurate information.

[redacted] *G0533,*

Former member, German PQE Committee, former Chairman, Afrikaans PQE Committee, present member, Dutch PQE Committee

---

The author replies:

I would like to express my appreciation to [redacted] [redacted] for his keen interest in my article on Xhosa

(CRYPTOLOG, 3rd Issue 1988, pp 14-15). I found his comments on the derivation of the Afrikaans language very useful in expanding on one of the points I had made. I do regret, however, that my reference to Afrikaans as a "bastardized version of 16th-century Dutch" triggered such an emotional response. I would like to assure Mr. Day that this comment was not meant to be perjorative. In this particular technical context, the word does not have a derogatory meaning as he states, but is merely descriptive, denoting the rapid simplification of Dutch in its new environment. (NB: Bastardize = to reduce from a higher to a lower state or condition.) In fact, the word is frequently used by scholars to classify Afrikaans and the linguistic process Dutch underwent as it evolved into the language we know today.

[redacted] passionate defense of Afrikaans clearly reflects a love of languages, which I share. I would like to thank him for providing me with the opportunity to comment on this subject.

[redacted] *PhD., G94*

CRYPTOLOG

welcomes your letters and comments

P.L. 86-36

~~FOR OFFICIAL USE ONLY~~

DOCID: 4010025

~~SECRET SPOKE~~

Reconstructed Signal Plan

Answers to TA Exercise on page 33

~~SECRET SPOKE~~

- Meaning of two "Q" signals:

+ QST = you change to ______________

+ QZY = I will change to ______________

- Four digit groups:

+ first digit indicates callsign table used (1 or 2)

+ second digit indicates column receiving callsign is taken from referenced table

+ third digit indicates column transmitting callsign is taken from referenced table

+ forth digit indicates frequency from frequency table (only one table used)

+ example - 1/472

1 = callsign table 1

4 = receiving callsign

7 = transmitting callsign

2 = frequency

1/472 = QEY DE CVQ 4700Kcs

Callsign tables:

Table-1

1	2	3	4	5	6	7	8	9	0
Z	B	S	Q	U	A	C	J	X	D
R	T	W	E	G	L	V	M	F	K
I	N	O	Y	H	P	Q	E	Z	R

Table-2

1	2	3	4	5	6	7	8	9	0
F	B	K	O	G	D	N	J	P	S
M	L	R	A	H	U	X	Q	W	M
E	T	Z	V	P	Y	O	C	I	G

Frequency table:

1	4450Kcs
2	4700Kcs
3	4825Kcs
4	4900Kcs
5	4975Kcs
6	5600Kcs
7	5775Kcs
8	5900Kcs
9	6150Kcs
0	6300Kcs

~~SECRET SPOKE~~

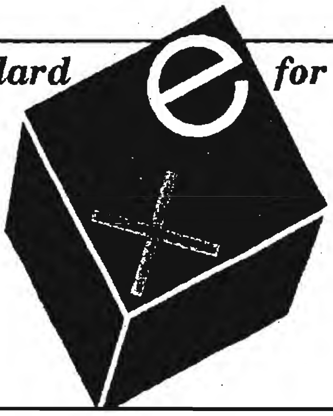
~~SECRET SPOKE~~

~~HANDLE VIA COMINT CHANNELS ONLY~~

Hardware Review

Defining a new standard for workstation computing

The NeXT Computer



[redacted] R535

P.L. 86-36

The NeXT Computer is the product of Steve Job's endeavors since leaving Apple Computer. It was designed after interviews with numerous universities and research organizations on what their needs and desires were for workstation computing. Initially it was to be offered only to educational institutions; arrangements have since been made to market it commercially through Businessland Computers.

The computer embodies new and diverse technologies which have the potential to define a new standard for high performance, low-cost, interactive workstation computing. In sum, it appears to offers a refreshing alternative to many workstation solutions which seem to stress only computational performance.

This paper discusses the important features of the NeXT Computer and will attempt to assess the impact of this technology on Agency applications. It will present the important hardware features that were designed into the architecture and how they could be exploited; it will the describe robust and dynamic software environment which appears at first glance to be superior to others currently on the market today; and finally, it will speculate on the significance of this technology for NSA computing architecture.

HARDWARE

At first glance, the NeXT Computer looks just like any other UNIX-based technical workstation. It consists of a 25 Mhz. Motorola 68030 CPU with a Motorola 68882 floating point unit, an Ethernet interface with TCP/IP and Network File System (NFS), 4-6 Megabytes of RAM memory, 32-bit NuBus system bus with 4 slots, a high resolution (1120x832), 17", 2-bit deep monochrome display, SCSI peripheral interface, 5.25 winchester disk, and keyboard with mechanical mouse. Though there are many vendors who sell systems with the same basic components, they do not offer the additional hardware features that the NeXT machine provides. These features allow for significant functionality not typically included by competitors in standard units.

MAGNETO-OPTICAL DISK DRIVE

The NeXT Computer incorporates a magneto-optical disk drive which provides a removable optical disk with up to 256 megabytes of data. This is the first instance of optical technology incorporated into the base platform of workstations. The drive is slow by comparison to winchester disk technology, 96-ms average seek time as compared to 18-ms average times for 5.25 winchester drives, but the increased capacity on a random access removable medium is very impressive. The complete operating system, very rich with bundled software is available on one optical disk. The disks are rather expensive, about $50.00 per disk, but that will probably decline as the device becomes more popular.

INTEGRATED AUDIO

It provides for high quality audio input and playback, with a built-in microphone jack that accepts a high impedance microphone signal. The microphone jack is connected to an analog-to-digital converter known as the CODEC. The CODEC converter produces 8-bit mulaw-encoded samples at a sample rate of 8 kHz. CODEC input is roughly equivalent to telephone quality speech and is suitable for a number of sound applications such as voice mail where limiting the size of sound objects is important.

The CODEC mu-law encoding is used to save storage space required for storing sound. The mu-law encoding allows a 12-bit dynamic range to be stored in only 8-bits. The 8-bit mu-law encoded sample will yield the same amplitude resolution as a linear 12-bit sample.

The NeXT Computer also integrates a Digital Signal Processor (20 MHz Motorola DSP56001) to process high-quality stereo sound. Its primary function is to minimize system overhead while processing sound, but it can also be programmed to process any type of digital data including signal filtering or image data. The DSP can record and play back up to 44.1 kHz samples per second in both the left and right channels with 16-bit quantization. The DSP can also process 24-bit image data and is able to perform multiple arithmetic functions on data within one instruction cycle of the DSP. Applications which require real-time data capture, such as voice input, can be effectively addressed with this processor.

The types of sound format currently supported under version 0.9 of the operating system are:

16-bit Linear, mono or stereo, 22.5 kHz or 44.1 kHz
16-bit linear, mono 8 kHz
8-bit mu-law, mono 8 kHz
8-bit linear, mono 22.5 kHz
DSP load image

For sound playback, the system contains a speaker built into the NeXT monitor with left and right channel line-out jacks on the back of the monitor. Sounds are sent to both the built-in speaker and to the stereo jacks during playback.

VLSI I/O PROCESSORS

The custom VLSI processors handles I/O between the different system components. NeXT studied workstation architectures and discovered that one significant limitation was in data throughput. In most workstations the CPU is interrupted to perform I/O operations on the system bus, including network transfer and storage medium interactions. In I/O-bound applications this becomes a serious performance problem which vendors usually address by higher bandwidth busses, faster CPU clock cycles and somewhat faster I/O controllers.

While incremental gains are made it still can be a bottleneck, especially when considering signal or image processing applications. NeXT decided that a major emphasis would be to significantly improve sustained system throughput. To accomplish this within the defined architecture required two special I/O processors whose function was to offload as much I/O from the CPU as possible. These two custom VLSI chips manage the SCSI interface, the optical disk drive (including error-correction logic), the serial ports and the Ethernet transfers.

To make efficient use of these components required custom DMA hardware between the CPU board and the I/O processors. There are currently 12 DMA channels on the CPU board to facilitate I/O throughput between devices and processors. To attain designed performance rates required enhancing the RAM memory to CPU data transfer rates. The NeXT computer can perform these transfers in burst mode, twice as fast as other Motorola CPU-based systems.

ERGONOMIC FEATURES

NeXT was very conscious about the system size and ergonomic features. The system consists of a 1-foot cube, a 17" monitor and a keyboard with a mechanical mouse. The system has one power plug which takes standard 110-120V power and is sufficient to supply the complete system. There is one 10 ft. cable from the system cube to the monitor and one cable from the monitor to the keyboard. The cube to monitor cable provides power, the video signal and audio signals for speaker output and microphone input. The limited number of cables makes for quick and easy installation. The keyboard has keys to control power to the system, video display intensity (lighten and darken) and audio volume. This allows all system interaction to take place from the keyboard and for the remoting of the cube away from the desktop if that is desired.

The hardware components are very cleanly integrated into a compact yet expandable cube. Few systems available today offer the powerful compute platform with the compact packaging that NeXT provides. The ultimate criterion will be the effectiveness of the software interface in developing applications which advantage of the unique hardware.

THE NeXT SOFTWARE ENVIRONMENT

The software environment provides an exciting first glance at the future for operating systems and the users' interaction with the system. NeXT has pushed the state of the industry by providing one of the first commercial offerings of the MACH operating system. It includes several important third party development tools and a user interface with development toolkits to assist in faster application development. The net effect is a dynamic, extensible, and robust environment which will appeal to both programmers and users.

We will discuss the software in three phases: the Mach operating system, the NextStep environment, and two of the application toolkits which are available for development.

THE MACH OPERATING SYSTEM

The Mach operating system was developed in 1986 at Carnegie Mellon University. It was originally intended as a multiprocessor operating system compatible with Berkeley 4.3 UNIX. It has received significant backing from DARPA (Defense Advanced Research Projects Agency), the major funding source for this effort. It also developed a mechanism for making extensions to UNIX without adding to the kernel. This allows for a smaller kernel base with system-specific functions executing outside the UNIX kernel. It also provides support for sharing memory between processes. This significantly improves performance when multiple processes are accessing large volumes of data. In traditional UNIX these processes would be copying data into each processs' memory space before accessing. In Mach the data is copied once into memory and copied again only when one of the processes attempts to modify it. It has been claimed that Mach outperforms UNIX by 15 to 25 percent because of these modifications.

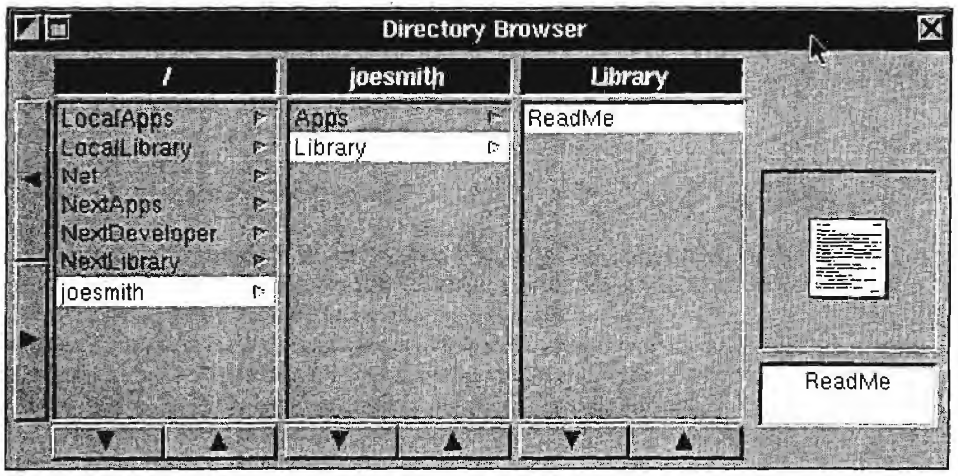
NeXT adopted Mach for several reasons. First, Mach is based on Berkeley UNIX which is widely used in the academic and scientific research communities. This allows theNeXT Computer access to many existing applications and experienced personnel in these communities. Second, NeXT sought a smaller kernel, eliminating functionality that was not pertinent for their environment. They also preferred having the ability to extend the system without adding to the kernel, which Mach allows. Third, with the major emphasis on maximizing data throughput, the ability to share memory was extremely attractive. And finally, support for multiprocessing provides NeXT with the possibility of adding more processors to its cube (there are three unused bus slots) without having to rewrite the operating system.

Even though the emphasis is placed on Mach, the user's perception will still be that the operating system is UNIX. The many traditional methods for UNIX hackers to interact are provided, including the structure and access to the file system. When interacting with the operating system the commands behave as a Berkeley UNIX system, including the network support for remote operations. NeXT was able to take advantage of the work pioneered at Sun Microsystems with the Network File System and the Yellow Pages utilities to ease the integration into existing network environments.

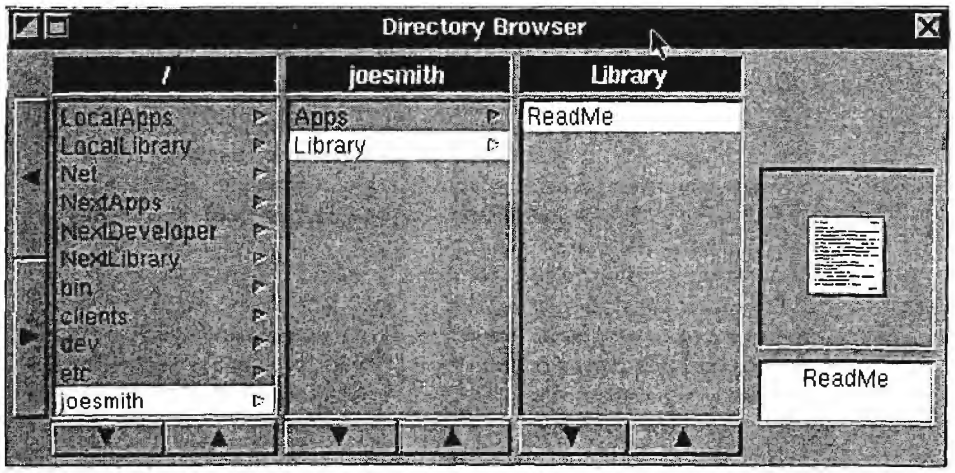
NeXTStep USER ENVIRONMENT

NeXT believes that powerful user interfaces are critical for systems of the nineties. Its proprietary interface called NextStep uses Display Postscript from Adobe systems for all text and graphics written to the display. It embodies a proprietary window system NeWS, similar to the Apple Macintosh in interaction and behavior, which is currently not compatible with X-windows or any postscript-based window system.

There is a complete graphical interface to the operating system and application toolkits. Applications are invoked by double clicking with the mouse on displayed icons and window manipulation is performed with the mouse pointing to specific areas in the window. The window system supports "hiding" applications which transforms the process into an icon on the lower part of the display where it continues to execute. The application can be recalled by double clicking the icon.



Simplified file system view (default)



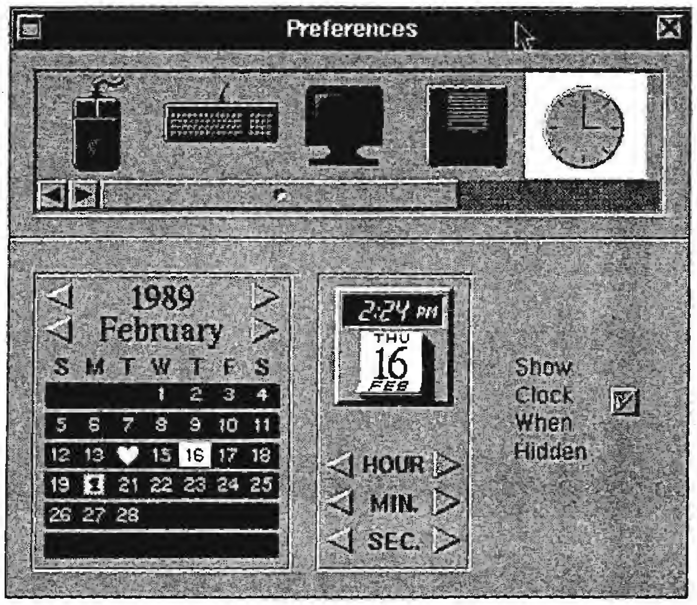
Browser with expert view enabled

Figure 1. Directories and Subdirectories side-by-side

The NextStep environment also provides a directory browser which displays the contents of directories and subdirectories side-by-side, as shown in Figure 1. The user selects an item with the mouse; if it is a subdirectory its contents will be presented in the column to the right of the current column. A graphical icon depicting the selected item is also displayed in the area immediately to the right of the columns. A user can easily use this tool to navigate through directories looking for files.

Windows are manipulated from the border area along the top edge. There can be several different icons within the border depending on the type of window used. These icons allow for functions like window resizing, moving the entire window pane to a new area on the screen, or terminating the window. Although it is not intuitive for novice users to immediately understand how to manipulate windows, it is easy to learn and users very quickly become adept at window interaction.

The NextStep environment with release 0.9 can be customized by users using the preference tool. This tool allows for setting user preferences for such things as key repeat rates, mouse scaling, etc., through a graphical



Show Clock When Hidden ✓

Preferences Panel

Figure 2. Example of Preference Tool

interface. Examples can be seen in Figures 2 and 3.

APPLICATION TOOLKITS

A major advantage of the NeXT software environment is the inclusion of toolkits to aid developers in building applications on the NeXT Computer. These toolkits consist of object-oriented function libraries which perform low-level object manipulation. Developers use these functions to define new objects required for the application.

This process takes advantage of the desirable features of object-oriented programming, namely extensibility and software reusability. NeXT provides an object-oriented language, Objective-C, as part of the standard environment to encourage development with object-oriented programming. Following are description of two toolkits provided.

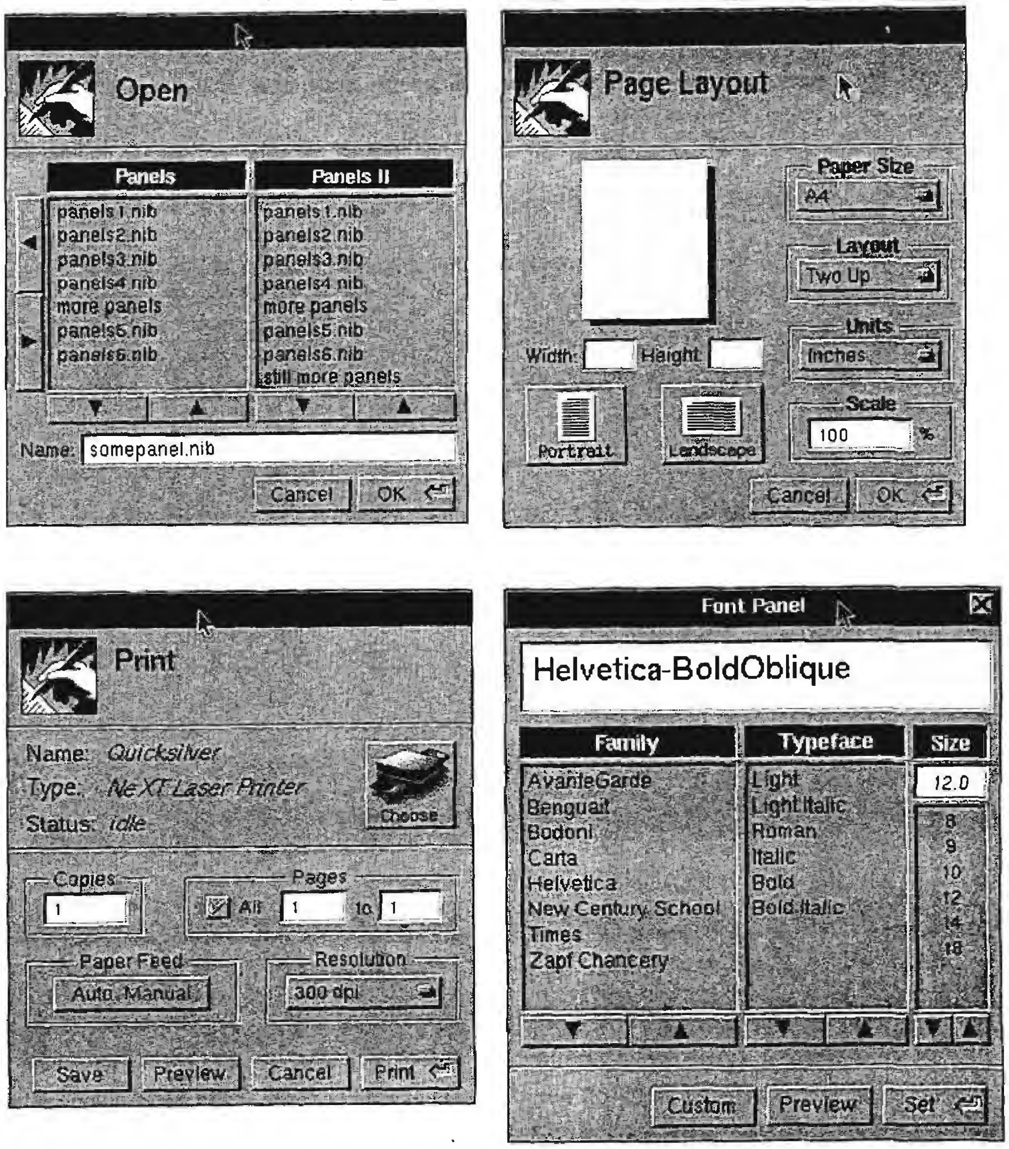
Interface Builder

The Interface Builder on the Next machine offers a powerful easy-to-use tool for constructing user interfaces. It allows one to generate the visual component of a user interface without typing a single line of code, yet it does not limit the complexity of interfaces that can be built.

An interface designer builds the visual component of the user interface by picking a graphics object from a set of available objects, and then dragging it to the interface building workspace. Once an object is in the workspace, one can modify it in various ways. For example an object can be resized, moved, copied, cut, pasted, grouped or ungrouped with other objects. Grouped objects can all be set to the same size and can be moved together as a single block. Tools are provided for aligning objects in rows, columns or matrices.

The interface designer selects graphics objects for the interface either from a set of defined objects or by creating a new graphics object. A system for creating and adding new graphical objects is provided, although the method is neither defined nor documented well. Examples of available graphics objects are: buttons, slide bars, text areas, boxes, switches, fields, windows, and panels. A pop-up menu can also be included in a user interface.

The Interface Builder also allows one to associate a sound or an icon with most graphics objects. This can be done by dragging the



Name: somepanel.nib

Paper Size: A4
Layout: Two Up
Units: Inches
Width: Height:
Scale: 100 %

Name: *Quicksilver*
Type: *NeXT Laser Printer*
Status: *idle*
Copies: 1
Pages: ☑ All 1 to 1
Paper Feed: Auto Manual
Resolution: 300 dpi

Helvetica-BoldOblique
Size: 12.0

Sample of standardized panels

Figure 3. Examples of Standardized Panels

sound or icon object onto the top of the target object. The system comes with a set of predefined sounds and icons which can be customized.

Once an interface has been built, it can be easily tested without the associated application. This allows one to work on the user interface without worrying about the actual application. A programmer and an end user can work together on the user interface for a project. They can completely build and test an interface before formally designing and coding the application. A complex user interface can be built in about five to ten minutes.

Graphical tools are provided for connecting an interface to an application. A programmer specifies how the interface and application components are associated. Then the Interface Builder uses this information to generate header and Objective-C codefiles. These files may be manipulated as the application is debugged. This toolkit (as well as others) automatically generate UNIX "make" files to allow for easy compilation of the finished product. This is extremely useful for developers as they maintain and rebuild software during the development process.

Sound Toolkit

The Sound Kit for analyzing and manipulating acoustical data consists of two Objective-C classes (Sound and SoundView) and a wide array of C functions to allow software developers to access the NeXT sound facilities with a minimum of effort. The Sound Kit software manages the details of OS communication, data access, and data buffering that are required during recording and playing sounds.

The Sound Kit provides the software developer with full access to the digitized samples of a sound object. Using the C functions provided, simple programs can be written to alter the pitch of a sound or change the playback speed of a sound. In addition, sounds can be played backwards, looped end-to-end, or divided into segments and reassembled in a different order. There are even software tools provided to digitally splice and mix together different sound objects.

The Sound class provides the facilities to create and manipulate sound objects. Sound objects are manipulated by sending messages to them. For example, the following lines of code are all that is necessary to create and record 5 seconds of sound:

```
/*  Create  a  new  sound  */
    id  mysound  =  [Sound  new];
    /*  Record  a  sound  */
    [mysound  record];
    /*  wait  5  seconds  */
    sleep(5);
    /*  Stop  recording  */
    [mysound  stop];
```

There are routines to record and play back a sound, as well as to read and write sound files. Facilities are also available for basic editing operations, such as deleting a portion of a sound or inserting part of one sound into another.

The SoundView class is provided to display sound objects in a predefined Soundview window. (Soundview windows can be built using the Interface Builder). Using the SoundView class, sound objects can be displayed as continuous waveforms, such as you would see on an oscilloscope, or as an outline of its maximum and minimum amplitudes. There are also methods provided for scrolling and zooming in and out of a displayed sound object, and selecting portions of a sound with the mouse.

Other Toolkits

Those are just two examples of the powerful toolkits which are available to developers with the NeXT Computer. Others provided are the text search toolkit which allows large bodies of text to be keyword indexed, searched or scanned for pattern matches; the Music Kit which allows for the generation and manipulation of synthesized music: the Application Kit to aid in object-oriented software development; and database management software (Sybase) which provides the tools to build and access relational databases.

CONCLUSIONS

The NeXT Computer has the potential to make significant advances in workstation computing within the Agency. It offers the most sophisticated integration of hardware and the most advanced software environments, all of which are included in the delivered system, in the smallest, cleanest packaging available today. Many of the important tools will be available on final operating system completion. Version 1.0 is expected about mid-summer.

How do we assess this technology? The general feeling of the programmers in R535 who have used the NeXT Computer is very positive. There is an extensive library of robust applications. We believe that rebuilding the prototypes developed on Suns and Apollos would be easier to do and functionally richer because we would begin building the application at a higher level. But to take advantage of these functionally rich libraries probably requires that applications be developed in Objective-C. This may be a problem for those without experience programming with object-oriented languages, though it is easy to learn in the NeXt step environment. A second potential problem might be performance-related if the application is not designed with object-oriented principles. The NeXT software environment continues to evolve, and with that evolution comes changes to the system. Applications being developed with the current 0.9 software release will not be object-code compatible with software release 1.0.

Software releases after 1.0 are supposed to be object-code compatible with applications developed at the 1.0 release. Developers should be aware of that if developing at 0.9.

R535 believes the NeXT Computer is appropriate for many types of applications at the Agency. It clearly competes with Apollo and Sun workstations for applications that do not require color displays. The tools for managing and searching large volumes of text, powerful database support and graphical tools for analysis make it appropriate for most analytic and reporting functions. The DSP processor and audio capabilities with the processing power available make it ideal for signals processing and transcription operations. The numerous tools for the software developer, including automatic generation of "make" files, make it appropriate for software development and maintenance. The unique capabilities it offers allows for innovative concepts and ideas to be explored from the research perspective. For these reasons, the real usefulness of the NeXT Computer has yet to be discovered. Interest in the Agency is growing. In the near future we will have a much better idea of how effective these machines can be within the Agency computing architecture.

REFERENCES

Denning, Peter J. and Karen A Frenkel, "A Conversation with Steve Jobs", *Communications of the ACM*, April 1989, Volume 32, No. 4.

Fisher, Sharon, "Mach, The New UNIX?", *UNIXWORLD*, March 1989, vol. VI, No. 3.

Thompson, Tom and Nick Baran, "The NeXT Computer", *BYTE*, November 1988, Vol. 13, No. 12.

NeXT 0.9/1.0 Release Description, April 1989.

BULLETIN BOARD

WISE NATIONAL CONFERENCE

The Interagency Committee for Women in Science and Engineering will hold its ninth annual National Training Conference Monday through Thursday, 26-29 March 1990 at the Westpark Hotel, Rosslyn, Virginia. This is an excellent opportunity to meet mathematicians, scientists and engineers in other government agencies, to network with them, and to gain some perspective. A highlight of the conference is an exhibit of science projects developed by 9th and 10th grade female students for science fairs held at their schools. These students represent the future. NSA women should encourage them to pursue careers in math, science, and engineering by taking the time to view their projects and yes, even, recruiting!

Concurrent workshops will be held Monday through Wednesday and a science tour is scheduled for Thursday morning. Cost for the full conference, which includes three luncheons and an evening reception, is $275. One-day fee is $135, and two-day fee is $150; both include all events of the day.

P.L. 86-36

For further information please call or write [redacted] Federal Women's Program Manager, D8, OPS 2B, 963-1103, or your local FWP representative.

FREE LISP MACHINE

P16 is offering a free LISP machine, available for the taking. This is a classic LMI Lambda system. The package consists of one processor, two terminals, a 6250 tape drive, software, documentation, and a support contract.

The machine can be viewed in the P1 machine room, accessed through 2N001 or 2N018. For information call Steve, 963-1103.

WRITING AND EDITING

CRYPTOLOG receives many brochures about conferences, seminars, and courses on writing and editing. The topics include manuals, online documentation, presenting science to the public, newsletters, slide and vuegraph presentations, proposals, procedures, pamphlets and brochures, proofreading, layout, interactive instruction, handbooks, multimedia presentations, and so on. They are available for consultation, as are the textbooks, references, handbooks, and conference proceedings in CRYPTOLOG's library.

The materials may be consulted in 2N018. For an appointment call the Editor, [redacted] 963-1103.

P.L. 86-36

~~FOR OFFICIAL USE ONLY~~

Review

Structure and Interpretation of Computer Programs

P.L. 86-36

Reviewed by: [redacted] *P16*

If anyone wishes to reexamine his preconceptions about the programming language LISP, I highly recommend the course *Structure and Interpretation of Computer Programs.* During 4-16 June I attended this course in Harpers Ferry, West Virginia. It featured videotaped lectures by MIT Professors Harold Abelson and Gerald Sussman on a variety of advanced programming techniques such as data-directed, object-oriented, stream-based, and logic programming. The lectures were complemented with lab work. Other issues covered during the course included the orders of growth for algorithms, contrasting models for the evaluation of procedures, procedural and data abstraction, the vanishing line between programs and data, and the inherent limitations of software engineering.

Structure and Interpretation of Computer Programs is a five-credit course at MIT required for all computer science and electrical engineering majors. It is based on an outstanding textbook of the same name written by Abelson and Sussman. It is significant that the MIT faculty selected LISP to illustrate many of the computer science concepts it considers essential for further study.

The National Cryptologic School offers this course both in an intensive two-week live-in format and in regularly scheduled daytime classes spread over an entire semester. I believe that instruction of this material in either configuration should be given highest priority because our computer scientists should be as forward looking as those at a leading technology-based university.

LISP AND SCHEME

LISP is bound to be a vital part of the future of computing. At the same time it is as old and resilient as FORTRAN. A testament to LISP's dynamism and continued importance is that Steven Jobs considered it worthwhile to bundle LISP with the system software of the NeXT Machine. Although *Structure and Interpretation of Computer Programs* is not a course in LISP, but in programming techniques, familiarization with LISP is a positive by-product of the course.

If employees' comfort level with LISP could be increased, this would enable us to adopt the latest research advances, particularly in the area of natural language processing, in which LISP plays a leading role, without worrying about lacking adequate support personnel. Since LISP is available on a multitude of platforms, including SUNs and PCs, the excessive maintenance cost of some specialized LISP machines is irrelevant.

Abelson and Sussman praised LISP (or the dialect SCHEME) as an ideal language for designing special-purpose languages appropriate to the task at hand. An example of this was that in the section on logic programming SCHEME was used to approximate PROLOG. They called this capability meta-linguistic abstraction. While LISP per se may not be well suited for any particular task, it provides the

~~FOR OFFICIAL USE ONLY~~

building blocks necessary to design higher-level languages optimized for a given task.

SCHEME, which is distributed by Texas Instruments, is described by the vendor as "a simple, modern LISP." It is also inexpensive and small enough to run on the PC. In the laboratory we used IBM-compatible laptop computers from Zenith. SCHEME is also available on the SUN and Macintosh.

One of the advantages of SCHEME over other dialects of LISP is that it enables the programmer to pass procedures as first-class objects to other procedures. This means that a procedure name can appear without any special syntax anywhere in another procedure to which this procedure name has been passed and still be applied properly. This is an elegant addition to the lambda notation--found in every version of LISP--which makes it possible for procedures to create and return other procedures. It also exemplifies SCHEME's simple design.

Key Concepts

A key concept in the course was the abstraction of procedures and data. Procedural abstraction is the notion that a procedure is just a name for a piece of code that performs some specific task. How this is done is intentionally concealed from the rest of a program so that higher level procedures invoking this procedure do not need to be concerned with the implementation details. There is an abstraction barrier between the code's implementation and its use. Data abstraction is analogous to procedural abstraction. It is defined as "a methodology that enables us to isolate how a compound data object is [illegible]d from the details of how it is constructed from more primitive data objects." (Abelson and Sussman, p. 74) Data constructors and selectors are used to keep the issues of data format and use separate. When data come in different formats they carry a label describing their format. Such data are called typed data.

Data abstraction provided a departure point for a discussion of data-directed programming. Data-directed programming is used to manipulate typed data with generic operators. Data-directed programs work with "a two-dimensional table that contains the possible operators on one axis and the possible types on the other axis." (Abelson and Sussman, p. 137) The body of the table contains the code corresponding to each pair of operator and type. Since this table guides the generic operator to the appropriate code, the format of the data is transparent to the user.

Abelson and Sussman illustrated data-directed programming by designing a program to perform arithmetic operations on complex numbers that were alternatively stored according to the polar or rectangular coordinate systems. The complex number package was one component of a large system for symbolic mathematics, including symbolic differentiation, that was examined in detail during the course. The symbolic mathematics system gave concrete expression to a multitude of concepts besides data-directed programming. For example, it was shown that recursion made it possible for the coefficients of polynomials to be themselves polynomials or complex numbers without disturbing the generic operators for manipulating mathematical expressions.

Object-oriented programming is ideal for modeling situations that can be described by a collection of objects that can be called upon to perform various actions. This is done by passing messages to the objects requesting that they perform such actions. The code for performing the actions is contained in the objects themselves, thereby erasing the line between procedures and data. Objects also contain local state variables that describe their condition at a given moment. This raises another issue: "how a computational object can change and yet maintain its identity." (Abelson and Sussman, p. 168)

Much as signal processing focuses on signal flow, so does stream-based programming or stream processing focus on data flow. A stream is a theoretically infinite "sequence of data objects." (Abelson and Sussman, p. 242) Stream processing gives us an alternative to conventional algorithms, which concentrate "on the order of events in a computation, rather than on the flow of data." (Abelson and Sussman, p. 245) From the perspective of stream processing, many computations can be reduced to a series of processes, specifically enumeration, filtering, mapping, and accumulation. Most programs combine these processes blindly, thus obscuring "the commonality that underlies many typical computations." (Abelson and Sussman, p. 243) By using this approach, the programmer can reduce significantly the amount of code needed for a set of problems and make the necessary code much more understandable.

Abelson and Sussman gave logic programming and PROLOG mixed reviews. They believe that declarative programming (where the programmer describes *what* he wants done, not *how* he wants it done) is an important paradigm, but that implementations of logic programming such as PROLOG have several defects. Some of the defects are restrictions on the placement of *not* clauses, infinite loops, and the closed world assumption that a program is aware of all information relevant to a query. The two professors held that *not* clauses were only useful as a filter to screen the possible answers already generated by the preceding clauses in a logical rule. They could not be placed in the first position among a series of clauses because they would not generate answers themselves and consequently would ensure that the rule would never return a positive response. The professors also maintained that a commutative rule (e.g., if X is married to Y then Y is married to X) combined with a fact to trigger the rule (e.g., Minnie is married to Mickey) would cause an endless loop.

Several reservations should be stated about this critique of PROLOG. Some believe that MIT is biased against PROLOG because it wasn't developed there, although LISP was. The infinite loops were confirmable in a commercial version of PROLOG, but they were not confirmed in a PROLOG-based knowledge engineering tool. The same tool provided a flag for the user to select either an open or closed world assumption. The approximation of PROLOG in SCHEME was much slower than PROLOG itself and could not eliminate duplicate responses. Incidently, Abelson and Sussman acknowledged that some excellent compiler work had gone into the design of PROLOG. However, the approximation was good enough to prove the point about LISP's capacity for meta-linguistic abstraction.

Along with abstraction barriers and programming paradigms, another theme was models for the evaluation of functions. Initially, the substitution model sufficed because there was no need to introduce an assignment operator into the language before the discussion of object-oriented programming with its requirement for local state variables, and thus assignment. The substitution model can be summarized by this rule: "to apply a compound procedure to arguments, evaluate the body of the procedure with each formal parameter replaced by the corresponding argument." (Abelson and Sussman, p. 184)

Once an assignment operator had been introduced, though, the substitution model broke down and had to be replaced by the environment model of evaluation. According to the environment model, expressions are evaluated with respect to the bindings in a series of frames called an environment. In a hierarchy of frames, bindings are inherited at lower levels, but they can be reassigned there without affecting the bindings in higher level frames. This permits assignment and local state, with the accompanying complications.

Abelson and Sussman generated some controversy by criticizing software engineering for promoting inflexibility. They stated that the product of software engineering is an elaborate edifice that performs the assigned task but cannot be easily modified to perform other tasks. They also maintained that it is impossible for a program module to serve more than one purpose under the software engineering approach, although this would be highly efficient. Despite these criticisms, however, Abelson and Sussman implicitly endorsed top-down design--a key characteristic of software engineering--by decomposing problems and referring to as yet non-existent code as "wishful thinking."

REACTIONS TO THE COURSE

Reactions to the course among fellow students varied widely. There were three main criticisms:

- The prerequisites for the course should have stated that to be successful, one needed a strong foundation in LISP and mathematics at least through one semester of Calculus;
- The material was too much to cover in two weeks; and
- The course was unrelated to work responsibilities.

A crash tutorial in LISP was offered beforehand to students without a background in LISP, but I believe that those with strong backgrounds in LISP were at an advantage. There is some dispute about this, though, because some students without any background in LISP ended up excelling, while others already proficient in LISP used their knowledge as a crutch to avoid learning the material as presented. In any event, the initial advantage eventually dissipated.

As for math, it certainly would be helpful to review or become familiar with the basics of differentiation, logarithms, and the conversions between the polar and rectangular coordinate systems beforehand since the professors relied on examples involving this level of math to illustrate many important principles. Whether calculus should be an absolute prerequisite is open to debate because arguably symbolic differentiation, for example, was only a means to explain a concept, not an end in itself.

That the material was too much to cover in two weeks is indisputable. By the second week, many of us displayed various forms of burn-out. I take exception, however, to the criticism that the course was unrelated to work responsibilities. Even if one does not program in LISP, many of the concepts and techniques presented in the course can be transferred to other languages and to a wide range of applications. But if these techniques are not being used at work, then that, along with an antagonism to LISP, is precisely what we should seek to change.

Endnotes

Abelson, Harold and Sussman, Gerald Jay, *Structure and Interpretation of Computer Programs*, MIT Press and McGraw Hill Book Company, 1985.

CLA

ESSAY CONTEST

closes COB 22 April 1990

START WRITING NOW! DON'T DELAY!

Money prizes go to the first, second, and third place winners.
Honorable mentions may also be awarded

RULES:

- The contest is open to Green and Orange badge holders.
- Essays must have language or linguistics as a major theme.
- Essays may be unclassified, or classified no higher than TSC; compartmented submissions cannot be accepted.
- While topics pertaining to language as it relates to culture, business, academic pursuits and other general themes are eligible, greater weight will be given to essays more closely pertaining to NSA applications.

Submit your entries to:

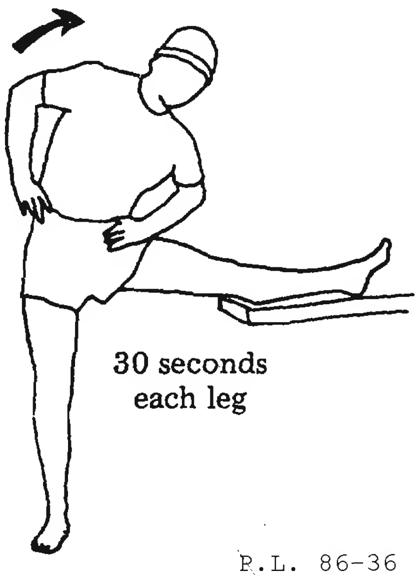
CLA Essay Contest
Attn: [redacted] A2405, Ops 2B, Room 2B5110.
963-3863

P.L. 86-36

~~FOR OFFICIAL USE ONLY~~

TA EXERCISE:

RECONSTRUCT THE SIGNAL PLAN



P.L. 86-36

[redacted]

~~(SC)~~ During the early 1960's, a Viet Cong (Vietnamese Communist) radio group operating in what was then the Eastern Nam Bo Region of South Vietnam, near the Cambodian border, used a signal plan that was different from other Vietnamese Communist (VC) communications in South Vietnam. The control authority was located close to the VC Central Office for South Vietnam and its outstations were located in the same general area. We could not make firm identifications of these entities, but we suspected that they may have represented logistic elements

~~(SC)~~ The group was active, or at least continuity was maintained, for about 18 months. During that period it did not undergo an overall communications change but random changes of call-ups and frequencies. Each time a change occurred one of two "Q" signals was sent followed by four digits; the first and second digits were always separated by a slash. Finally we recovered the system used to allocate the callsigns and frequencies.

(U) Shown below are examples of the "Q" signals and four-digit groups followed by the call-ups and the frequencies used after they were sent by control or outstations.

(U) See if you can recover the system.

~~(SC)~~ Control changes to call-up and frequencies listed below after sending QZY X/XXX:

QZY	1/472	QEY	DE	CVQ	4700Kcs
QZY	2/615	DUY	DE	FME	4975Kcs
QZY	1/747	ZRI	DE	QEY	5775Kcs
QZY	1/934	XFZ	DE	SWO	4900Kcs
QZY	2/128	FME	DE	BLT	5900Kcs
QZY	2/393	KRZ	DE	PWI	4825Kcs
QZY	1/281	BTN	DE	JME	4450Kcs
QZY	2/586	GHP	DE	JQC	5600Kcs
QZY	2/939	PWI	DE	KRZ	6150Kcs
QZY	2/240	BLT	DE	OAV	6300Kcs
QZY	1/352	SWO	DE	UGH	4700Kcs

~~(SC)~~ Control changes after receiving QST X/XXX:

QST	2/476	OAV	DE	NXO	5600Kcs
QST	1/561	UGH	DE	ALP	4450Kcs
QST	1/629	ALP	DE	BTN	6150Kcs
QST	2/757	DUY	DE	GHP	5775Kcs
QST	2/845	JQC	DE	OAV	4975Kcs
QST	1/128	ZRI	DE	BTN	5900Kcs
QST	1/373	SWO	DE	CVQ	4825Kcs
QST	1/842	ZRI	DE	QEY	4700Kcs
QST	2/963	PWI	DE	DUY	4825Kcs
QST	2/020	SMG	DE	BLT	6300Kcs

~~(SC)~~ Outstation changes after sending QZY X/XXX:

QZY	1/016	DKR	DE	ZRI	5600Kcs
QZY	2/602	DUY	DE	SMG	4700Kcs
QZY	1/157	ZRI	DE	UGH	5775Kcs
QZY	2/579	GHP	DE	NXO	6150Kcs
QZY	1/421	QEY	DE	BTN	4450Kcs
QZY	1/083	DKR	DE	JME	4825Kcs
QZY	2/458	OVA	DE	GHP	5900Kcs
QZY	2/970	PWI	DE	NXO	6300KCS
QZY	1/292	BTN	DE	XFZ	4700Kcs
QZY	2/746	NXO	DE	OAV	5600Kcs
QZY	1/324	SWO	DE	BTN	4900Kæcs

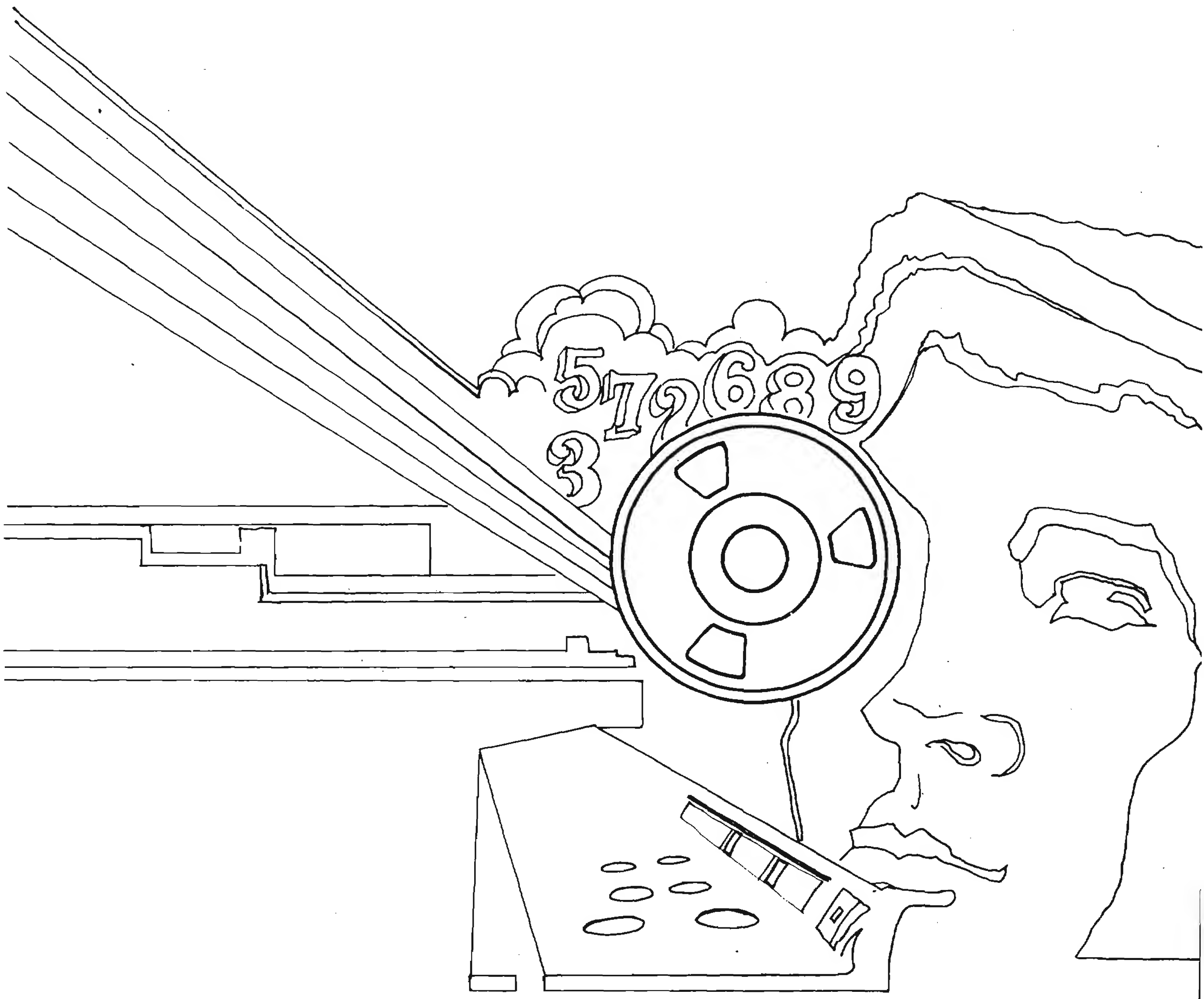
~~(SC)~~ Outstation changes after receiving QST X/XXX:

QST	2/370	KRZ	DE	NXO	6300Kcs
QST	1/014	DKR	DE	ZRI	4900Kcs
QST	2/605	DUY	DE	SMG	4975Kcs
QST	1/261	BTN	DE	ALP	4450Kcs
QST	2/486	OAV	DE	JQC	5600Kcs
QST	1/090	DKR	DE	XFZ	6300Kcs
QST	1/763	CVQ	DE	ALP	4825Kcs
QST	2/148	FME	DE	OAV	5900Kcs
QST	1/905	XFZ	DE	DKR	4975Kcs
QST	1/839	JME	DE	SWO	6150Kcs
QST	2/397	KRZ	DE	PWI	5775Kcs

The reconstructed signal plan appears on page 20.

DOCID: 4010025

~~SECRET~~



~~THIS DOCUMENT CONTAINS CODEWORD MATERIAL~~

~~SECRET~~

~~NOT RELEASABLE TO CONTRACTORS~~